UNIVERSIDADE FEDERAL DE SÃO CARLOS
DEPARTAMENTO DE COMPUTAÇÃO

Construção de Compiladores

Documentação Interna - T1

Áquila Oliveira, 759313

Carlos Eduardo Fontaneli , 769949

Ingrid Lira dos Santos, 790888

São Carlos - SP

29 de maio de 2023

1 Documentação Interna

Documentação interna do arquivo LALexer.g4

Início do analisador lexico da linguagem LA

```
lexer grammar LALexer;
```

O programa, por padrão, inicia com uma declaração, cuja palavra-chave é 'algoritmo', e um corpo, finalizando com a palavra-chave 'fim algoritmo'

```
PROGRAMA: DECLARACOES 'algoritmo' CORPO 'fim_algoritmo';
fragment DECLARACOES: (DECL_LOCAL_GLOBAL)*;
fragment DECL_LOCAL_GLOBAL: DECLARACAO_LOCAL | DECLARACAO_GLOBAL;
```

Definição de padrão para números inteiros e números reais

```
NUM_INT: ('0'..'9')+;
NUM_REAL: ('0'..'9')+ PONTO ('0'..'9')+;
```

Abaixo, temos o padrão para as expressões literais, ou cadeias

```
CADEIA: ASPAS ('a'..'z' | 'A'..'Z' | ' ' | '.' | '/' | DELIM | '?' | '<'
    | '-' | '>' | '+' | ',' | '(' | ')' | '!' | '$' | '=' | ';')* ASPAS;
ASPAS: '"';
```

Segue a lista de palavras chaves:

```
ALGORITMO: 'algoritmo';
FIM_ALGORITMO: 'fim_algoritmo';
FACA: 'faca';
DECLARE: 'declare';
CONSTANTE: 'constante';
TIPOO: 'tipo';
COCHESQ: '[';
COCHDIR: ']';
LITERAL: 'literal';
INTEIRO: 'inteiro';
REAL: 'real';
LOGICO: 'logico';
REGISTROO: 'registro';
FIMREGISTRO: 'fim_registro';
PROCEDIMENTO: 'procedimento';
FIMPROCED: 'fim_procedimento';
FUNCAO: 'funcao';
```

```
18 FIM_FUNCAO: 'fim_funcao';
19 VAR: 'var';
20 LEIA: 'leia';
21 ESCREVA: 'escreva';
22 SE: 'se';
23 ENTAO: 'entao';
24 FIM_SE: 'fim_se';
25 SENAO: 'senao';
26 CASO: 'caso';
27 FIM_CASO: 'fim_caso';
28 PARA: 'para';
29 FIM_PARA: 'fim_para';
30 ENQUANTO: 'enquanto';
31 FIM_ENQUANTO: 'fim_enquanto';
32 ATE: 'ate';
33 RETORNE: 'retorne';
34 VERDADEIRO: 'verdadeiro';
35 FALSO: 'falso';
36 SEJA: 'seja';
37 NAO: 'nao';
```

Como o objetivo do trabalho, de maneira geral, é apenas apresentar as palavras-chave de cada um dos respectivos tokens, utilizamos "fragment"antes de grande parte dos padrões utilizados. Para que eles sirvam como uma variável auxiliar que permitam que somente as palavras-chave esperadas retornem durante a execução do programa. Vale ressaltar que o corpo contém toda a lógica de programação da linguagem LA

```
1 fragment CORPO: (DECLARACAO_LOCAL)* (CMD)*;
2
3 fragment DECLARACAO_LOCAL: 'declare' VARIAVEL
4                 | 'constante' IDENT DELIM TIPO_BASICO '='
     VALOR_CONSTANTE
5                 | 'tipo' IDENT DELIM TIPO;
6 fragment DECLARACAO_GLOBAL: 'procedimento' IDENT ABREPAR (PARAMETROS)?
     FECHAPAR (DECLARACAO_LOCAL)* (CMD)* 'fim_procedimento'
7                  | 'funcao' IDENT ABREPAR (PARAMETROS)? FECHAPAR DELIM
     TIPO_ESTENDIDO (DECLARACAO_LOCAL)* (CMD)* 'fim_funcao';
8
9 OP_UNARIO: '-';
```

```
10
11 fragment VARIAVEL: IDENTIFICADOR (SEPAR IDENTIFICADOR)* DELIM TIPO;
12 fragment IDENTIFICADOR: IDENT (PONTO IDENT)* DIMENSAO;
13 fragment DIMENSAO: ('[' EXP_ARITMETICA ']')*;
14 fragment TIPO: REGISTRO | TIPO_ESTENDIDO;
15 fragment TIPO_BASICO: 'literal' | 'inteiro' | 'real' | 'logico';
16 fragment TIPO_BASICO_IDENT: TIPO_BASICO | IDENT;
17 fragment TIPO_ESTENDIDO: (CIRCUNF)? TIPO_BASICO_IDENT;
18 fragment VALOR_CONSTANTE: CADEIA | NUM_INT | NUM_REAL | 'verdadeiro' | '
    falso';
19 fragment REGISTRO: 'registro' (VARIAVEL)* 'fim_registro';
20
21 fragment PARAMETRO: ('var')? IDENTIFICADOR (SEPAR IDENTIFICADOR)* DELIM
    TIPO_ESTENDIDO;
22 fragment PARAMETROS: PARAMETRO (SEPAR PARAMETRO)*;
```

Padrões para os comandos do programa

```
1 fragment CMD: CMD_LEIA | CMD_ESCREVA | CMD_SE | CMD_CASO | CMD_PARA |
    CMD_ENQUANTO | CMD_FACA | CMD_ATRIBUICAO | CMD_CHAMADA | CMD_RETORNE;
2 fragment CMD_LEIA: 'leia' ABREPAR ((CIRCUNF)? IDENTIFICADOR (SEPAR (
    CIRCUNF)? IDENTIFICADOR)*) FECHAPAR;
3 fragment CMD_ESCREVA: 'escreva' ABREPAR EXPRESSAO (SEPAR EXPRESSAO)*
    FECHAPAR;
4 fragment CMD_SE: 'se' EXPRESSAO 'entao' (CMD)* ('senao' (CMD)*)? 'fim_se
    ';
5 fragment CMD_CASO: 'caso' EXP_ARITMETICA 'seja' SELECAO ('senao' (CMD)*)
    ? 'fim_caso';
6 fragment CMD_PARA: 'para' IDENT '<-' EXP_ARITMETICA 'ate' EXP_ARITMETICA
     'faca' (CMD)* 'fim_para';
7 fragment CMD_ENQUANTO: 'enquanto' EXPRESSAO 'faca' (CMD)* 'fim_enquanto'
    ;
8 fragment CMD_FACA: FACA (CMD)* 'ate' EXPRESSAO;
9 fragment CMD_ATRIBUICAO: (CIRCUNF)? IDENTIFICADOR '<-' EXPRESSAO;
10 fragment CMD_CHAMADA: IDENT ABREPAR EXPRESSAO (SEPAR EXPRESSAO)*
    FECHAPAR;
11 fragment CMD_RETORNE: 'retorne' EXPRESSAO;
12 fragment SELECAO: (ITEM_SELECAO)*;
13 fragment ITEM_SELECAO: CONSTANTES DELIM (CMD)*;
14 fragment CONSTANTES: NUMERO_INTERVALO (SEPAR NUMERO_INTERVALO)*;
```

```
15 fragment NUMERO_INTERVALO: (OP_UNARIO)? NUM_INT ('..' (OP_UNARIO)?
       NUM_INT)?;
16 fragment OP_RELACIONAL: '=' | '<>' | '>=' | '<=' | '>' | '<';
17
18 fragment EXP_ARITMETICA: TERMO (OP1 TERMO)*;
19 fragment TERMO: FATOR (OP2 FATOR)*;
20 fragment FATOR: PARCELA (OP3 PARCELA)*;
21 fragment PARCELA: (OP_UNARIO)? PARCELA_UNARIO | PARCELA_NAO_UNARIO;
22 fragment PARCELA_UNARIO: (CIRCUNF)? IDENTIFICADOR
23                 | IDENT ABREPAR EXPRESSAO (SEPAR EXPRESSAO)* FECHAPAR
24                 | NUM_INT
25                 | NUM_REAL
26                 | ABREPAR EXPRESSAO FECHAPAR;
27 fragment PARCELA_NAO_UNARIO: '&' IDENTIFICADOR | CADEIA;
28 fragment EXP_RELACIONAL: EXP_ARITMETICA (OP_RELACIONAL EXP_ARITMETICA)?;
29 fragment EXPRESSAO: TERMO_LOGICO (OP_LOGICO_1 TERMO_LOGICO)*;
30 fragment TERMO_LOGICO: FATOR_LOGICO (OP_LOGICO_2 FATOR_LOGICO)*;
31
32
33 fragment FATOR_LOGICO: NAO? PARCELA_LOGICA;
34 fragment PARCELA_LOGICA: ('verdadeiro' | 'falso') | EXP_RELACIONAL;
```

Operadores lógicos:

```
1 OP_LOGICO_1: 'ou';
2 OP_LOGICO_2: 'e';
```

Termos que não são reconhecidos pela linguagem LA e são ignorados. Espaços em branco, quebras de linha e tabulação também são ignorados.

```
1 IGNORAR: ('ã' | 'õ' | 'ç' | 'à' | 'á' | 'é' | 'è' | 'ó' | 'ò' | 'ú' | 'ù'
      | 'â' | ';' | 'í' | '!' | 'ê') -> skip;
2
3 ABREEE: '{ ';
4 ABRECHAVE: '{';
5 FECHACHAVE: ' }';
6
7 ESPACO: (' '|'\t'|'\r'|'\n') -> skip;
```

Operadores relacionais

```
1 IGUAL: '=';
```

```
DIFERENTE: '<>';
MAIORIGUAL: '>=';
MENORIGUAL: '<=';
MAIOR: '>';
MENOR: '<';
MAIS: '+';
OP_DIV: '/';
OP_MULT: '*';
OP1: (MAIS | OP_UNARIO);
OP2: (OP_MULT | OP_DIV);
OP3: '%';
```

Definição de padrão dos identificadores.

```
IDENT: ('a'..'z' | '_' | 'A'..'Z') ('a'..'z' | '_' | '0'..'9' | 'A'..'Z'
    )*;

FLECHA: '<-';
DELIM: ':';
ABREPAR: '(';
FECHAPAR: ')';
SEPAR: ',';
PONTO: '.';
CIRCUNF: '^';
EZINHO: '&';
SEQUENCIA: '..';
```

Símbolos que não são reconhecidos pelo sistema.

```
ERR_SIMBOLO_NAO_PERMITIDO: ('$' | '~' | '}');
```

Token pra identificação de comentário, comentário não fechado e cadeia de literal não fechada

```
COMENTARIO:'{' ~[\r\n{}]* '}' [\r]? [\n]? -> skip;

COMENTARIO_NAO_FECHADO: ('{' | '{ ') ~('\n'| '\r' | '}')* ('\n' | '\r');

CAD_LITERAL_NAO_FECHADA: '"' (~('\n'|'\''|'\\' | '"'))* ('\n' | '\r');
```